README – Control MKS SERVO57D with ROS 2 Jazzy on Raspberry Pi

This guide explains how to control a Makerbase MKS SERVO57D stepper-servo driver from a Raspberry Pi running Ubuntu 24.04 and ROS 2 Jazzy, using a custom ROS 2 node written in Python.

1. Prerequisites

Hardware

- Raspberry Pi 4 (or similar)
- microSD card with Ubuntu 24.04 (Noble) installed
- Makerbase MKS SERVO57D (RS485 version)
- USB–RS485 adapter connected to the Raspberry Pi
- Proper wiring between the RS485 adapter and the servo driver:
  - Adapter A / D+ → Driver A+ / D+
  - Adapter B / D– → Driver B– / D–

Make sure the motor is powered correctly and the RS485 port is used (not CAN).

Software on the Raspberry Pi

- Ubuntu 24.04 already installed
- ROS 2 Jazzy already installed and working

You should be able to run:

```
ros2 --help
```

Connect to the Raspberry Pi via SSH

- From your PC:

```
ssh horseshitbot@<PI_IP_ADDRESS>
```

2. Install Python dependencies on the Pi

On the Raspberry Pi (over SSH):

```
source /opt/ros/jazzy/setup.bash
sudo apt update
sudo apt install -y python3-pymodbus python3-serial
```

3. Create the ROS 2 workspace

Create a workspace named ROS2_HORSESHITBOT:

```
cd ~
mkdir -p ROS2_HORSESHITBOT/src
cd ROS2_HORSESHITBOT/src
```

4. Create the driver package

In the workspace src folder:

```
ros2 pkg create --build-type ament_python mks_servo_driver
```

This generates a Python ROS 2 package:

```
ROS2_HORSESHITBOT/
  src/
    mks_servo_driver/
      package.xml
      setup.py
      setup.cfg
      resource/mks_servo_driver
      mks_servo_driver/__init__.py
      test/...
```

5. Add the servo node (servo_node.py)

Move into the package's Python module directory:

```
cd ~/ROS2_HORSESHITBOT/src/mks_servo_driver/mks_servo_driver
nano servo_node.py
```

Paste your ROS 2 node code for the MKS SERVO57D into this file. The code should:
- Create a Modbus RTU client (using pymodbus)
- Configure the driver (work mode, enable)
- Expose a ROS 2 node that subscribes to cmd_speed (type std_msgs/msg/Int32)
- Send speed commands to the motor using the Modbus registers

Save and exit nano:
- Ctrl + O, Enter
- Ctrl + X

6. Configure setup.py

Go back to the package root:

```
cd ~/ROS2_HORSESHITBOT/src/mks_servo_driver
nano setup.py
```

Example setup.py content:

```
from setuptools import setup

package_name = 'mks_servo_driver'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
         ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
```

```
    zip_safe=True,
    maintainer='horseshitbot',
    maintainer_email='horseshitbot@todo.todo',
    description='ROS 2 driver for the MKS SERVO57D stepper-servo via Modbus RTU',
    license='MIT',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            # name         = package.module:function
            'servo57d_node = mks_servo_driver.servo_node:main',
        ],
    },
)
```

Save and exit.

7. Build the workspace

From the workspace root:

```
cd ~/ROS2_HORSESHITBOT
source /opt/ros/jazzy/setup.bash
colcon build
```

If the build succeeds, add the workspace setup to your shell configuration:

```
echo "source ~/ROS2_HORSESHITBOT/install/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

8. Run the driver node

Use two terminals (both SSH on the Pi).

Terminal 1 – start the node:

```
ssh horseshitbot@<PI_IP_ADDRESS>
source /opt/ros/jazzy/setup.bash
source ~/ROS2_HORSESHITBOT/install/setup.bash
cd ~/ROS2_HORSESHITBOT
ros2 run mks_servo_driver servo57d_node
```

Terminal 2 – send speed commands:

```
ssh horseshitbot@<PI_IP_ADDRESS>
source /opt/ros/jazzy/setup.bash
source ~/ROS2_HORSESHITBOT/install/setup.bash
```

Check that the node is visible:

```
ros2 node list
```

Expected:

```
/servo57d_node
```

Send speed commands:

```
ros2 topic pub /cmd_speed std_msgs/msg/Int32 "{data: 800}" --once
ros2 topic pub /cmd_speed std_msgs/msg/Int32 "{data: 300}" --once
```

ros2 topic pub /cmd_speed std_msgs/msg/Int32 "{data: 0}" --once

- data: 0 means stop
- data: N (0 < N ≤ 3000) sets the speed to N RPM (within the limits in your code).

9. Delete the workspace (optional)

If you want to erase this workspace and start over without removing ROS 2:

1. Remove the workspace:
   cd ~
   rm -rf ROS2_HORSESHITBOT

2. Edit ~/.bashrc to remove:
   source ~/ROS2_HORSESHITBOT/install/setup.bash

3. Reload and check ROS 2:
   source ~/.bashrc
   ros2 --help